Introduction to Blockchain Technology and Applications

Prof. Sandeep Shukla
Department of Computer Science and Engineering
Indian Institute of Technology-Kanpur

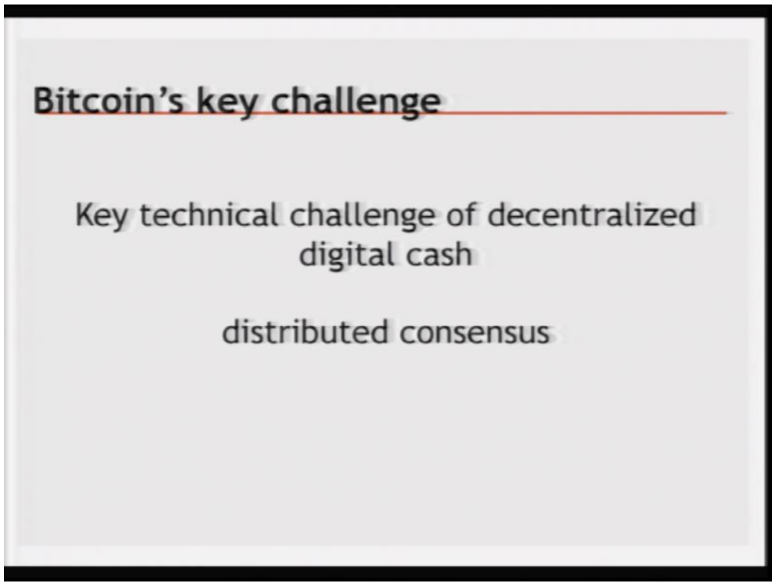
Lecture No. 05
Blockchain Technology and Applications

(Refer Slide Time: 00:14)

Distributed consensus

Welcome back. So, now as I said before that we will be discussing consensus mechanism in Bitcoin blockchain.

(Refer Slide Time: 00:22)



Bitcoin's key challenge

Key technical challenge of decentralized digital cash

distributed consensus

The main technical challenge, so digital cash has been thought about and many proposals have been made by computer scientists for almost 20 years before the 1089, when the Bitcoin blockchain was introduced, and that was the first cryptocurrency that we saw. So always there was this question of the consensus that bothered people. Now why is it important because when you create new money, or when you create a new transaction?

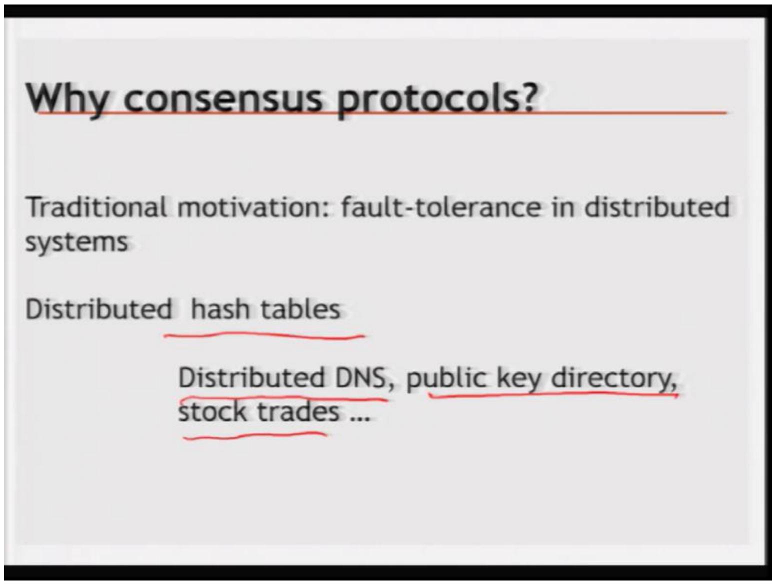
Somebody has to validate that this is the valid transaction. Somebody also has to check that you are not double spending so you acquired some money. So let us say 25 bitcoins, and you pay somebody 25 bitcoins and again, you pay the same 25 bitcoins to somebody else that should be disallowed. Now, with a central authority that is very easy to check, but with a decentralized system, it is much harder problem.

Because if one node here says that what you are doing is not acceptable, other nodes may say that what you are doing is acceptable. So, therefore, one has to decide that what should be the verdict of the entire network. Because somebody might be maliciously saying that what you are doing is not acceptable when you actually did perfectly correct thing. In another possibility is that we are the some of the good participants got fooled?

Because you did something wrong, but you actually fooled them or it could be that the number of malicious guys are going trying to say some bad transaction is good transaction like you have done double spending and then you have told your friends that you vote for me and you say that this is a good transaction that transaction gets into the blockchain and it becomes permanent and that is not good.

So, therefore, this consensus is a mechanism by which you say we or all of us agree to agree on something or that something is acceptable. So, in a distributed setting, consensus has always been a problem. And this problem has been looked into almost for 40 years.

(Refer Slide Time: 02:57)



Why consensus protocols?

Traditional motivation: fault-tolerance in distributed systems

Distributed hash tables

Distributed DNS, public key directory, stock trades ...

So, let us see what are the things that happens; now consensus as I said other than in case of blockchain how the consensus problem came about in the in the distributed computing community. So, in a distributed computing system you have distributed set of nodes and this set of nodes have to compute something. So, let us say they have to compute some function or they might have to compute that you know, some routing table or they may have to compute, they may have to together maintain information and information integrity.

These are the kind of things that distributed computing community worry about now to in order to have a distributed system world, many times people do what is called a leader election. So you have n number of nodes, and then you run a protocol that everybody engages in and eventually say, here is the node that will be the leader. So once I have done the leader election, I will listen to the leader throughout the computation.

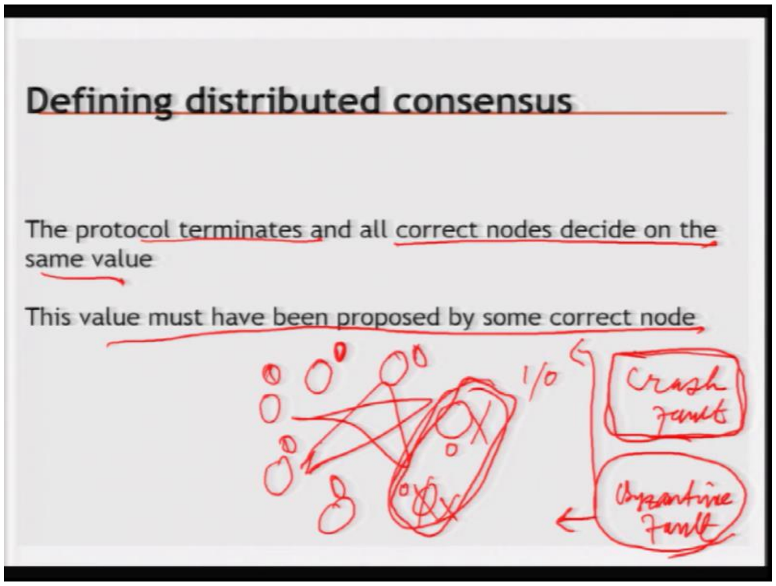
So that leader can then give you work that you do this amount of work you do this amount of work, and together you accomplish something. So that is basically called a leader election protocol. So similarly, be sometimes the distributed system has to do some voting. For example, for leader election also, you may have a voting based leader election. But sometimes you have to vote on a piece of data. And therefore, you may have to do something called a consensus.

So this problem has bothered distributed computing people for more than 40 years now. And some of the applications they had thought about even before blockchain is that I may have to create a hash table, but I did not want to be wanted to be maintained by just a single authority. I want this hash table maintained by everybody. So everybody will have a copy of the hash table. And this copy of the hash table should be the same.

If you have that, then you can have a distributed domain name service. Earlier I said that the distributed domain name service right now is kind of centralized. But if you have a distributed DNS, then you know, you are free from the dependence on a trusted party public key directory. So when you have public key remember like yesterday, we discussed that if you have public key, you know, telling everybody else that is it your public key.

And also convincing them that it is your public key requests digital certificates. Digital certificate is basically when a trusted third party tells the others that indeed it is your public key. Now, you can actually maintain a public key directory distributedly in a decentralized fashion through a distributed hash table if you can solve that problem. And then other places like stock trades and so on. So there this has been a problem distributed computing problem for a long time.

(Refer Slide Time: 06:06)



Defining distributed consensus

The protocol terminates and all correct nodes decide on the same value

This value must have been proposed by some correct node

So how do we define what the distributed consensus problem in a formal way, so what we say is that distributed consensus is a protocol that terminates and then all nodes that are not malicious decide on the same value and this value must be proposed by at least one correct node. So, let us say you have n nodes, nodes means these are computing nodes and they are connected by some topology here. And they are going to decide on some value.

So, everybody proposes a value for example, if it is a binary consensus, then everybody proposes either 1 or 0. So, everybody proposes something. And then you run the protocol at the end of the protocol, everybody should have, let us say among these, these 2 are bad factors. And they are good, they are proposing 0. Everybody else is proposing 1. Now, during the protocol exercise, these guys will not also listen to follow the rules of the protocol.

So they will try to send messages, they might try to confuse the others. But if the distributed consensus protocol is robust enough, that is, with the existence of this malicious guys doing malicious things. Eventually, all the good guys will decide on one because one was something that was proposed by one of them actually, in this case, in my example, all of them, but maybe this guy might have done something else.

But what they agree on is 1, then we know that we have reached a consensus. Now since one of the good guys also did 0. In this case, if everybody eventually agrees of 0 that would be also a correct outcome, because what it says is that the protocol terminates and all correct nodes decide on the same value and that value must have been proposed by some correct node. Now, the only complication here is that protocol design and does not know how many are going to be malicious node here.

Second thing he does not know is who are the malicious ones right. So therefore, the protocol has to be designed in such a way so that the no matter who are the malicious nodes, this consensus will eventually terminate and a consensus will be met and the consented value by the good nodes. Non malicious nodes should be chosen from the values that were given by pleased one of them. So anything that the malicious guys proposed as the value that should be chosen.

If none of the other good guys have proposed that value, then that should never be the end result of the consensus. So that is the idea. Now, it turns out that depending on what kind of maliciousness you are going to see, in these nodes, the problems gets different. So for example, in the very beginning, people started saying that, well, if the while the protocol is running, if one of the nodes crashes, so that is not really malicious, the node might crash for other reasons.

Even then the consensus problem becomes really hard. So it is so if the one of the node crashes, then also consensus problem becomes hard. So this is called a crash fault. So if we have a protocol that works, in spite of crash of some nodes, then we say it is a crash tolerant or crash fault tolerant protocol, but in case it is protocol fails that is it fails to terminate or it terminates but it gives incorrect consequences then we say that the protocol is not tolerant to crash fault.

Now, crash fault is the most benign kind of fault because the node that has that has faulted is not doing anything malicious, just crashed, that now, the one problem in all this in the decentralized setting is that other nodes cannot tell whether the node crashed or he is maliciously keeping quiet to create confusion. So therefore, we have to assume that the crash fault is could be a behavior of a malicious node or it could be really crashed.

A crash fault is the easiest of the maliciousness. The worst kind of maliciousness is called a Byzantine fault. So Byzantine fault basically means that the nodes that are malicious they are not only going to propose wrong values in the beginning, they are also going to not follow the rules of the protocol. And therefore they will be doing things that are not part of the protocol protocols. What is a protocol is a program for every node.

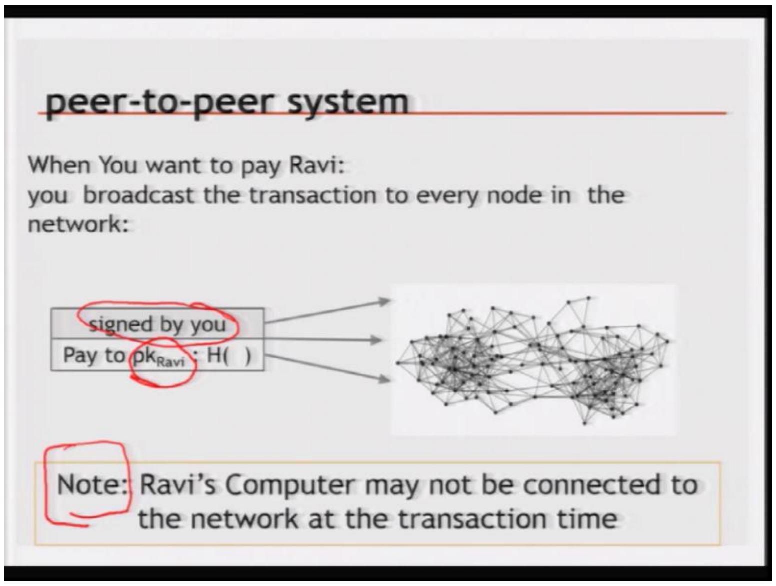
And the, what they and each program here could actually do some computation and communicate it with the other nodes. And then again, do some computation communicate with other nodes. This is what a protocol looks like, right? So a protocol, then obviously has a set of rules or the how the program is supposed to run. The malicious nodes may actually run a different program.

Maybe it is because they have been infected by a malware which has a different program, or it may actually do various things like exchange messages between the other malicious nodes and not ever communicate with others. It can do all kinds of things. So the question is what do, we want in case of a Bitcoin case for sample we also have to worry about nodes that will work maliciously.

For example, they will validate transactions which are not supposed to be valid or they might actually not valid good transactions, or they might actually create blocks and that those blocks may be having invalid transaction in them and broadcast that block for saying that this is the next block. So they can do all kinds of things. So that in that case also which block has to be eventually added to the network.

Next to the blockchain next is a consensus problem. And the protocol should be robust enough that even if some of the nodes act like that, we have the ability to eventually have a correct block added to the to the blockchain.

(Refer Slide Time: 12:57)



peer-to-peer system

When You want to pay Ravi:
you broadcast the transaction to every node in the network:

Note: Ravi's Computer may not be connected to the network at the transaction time

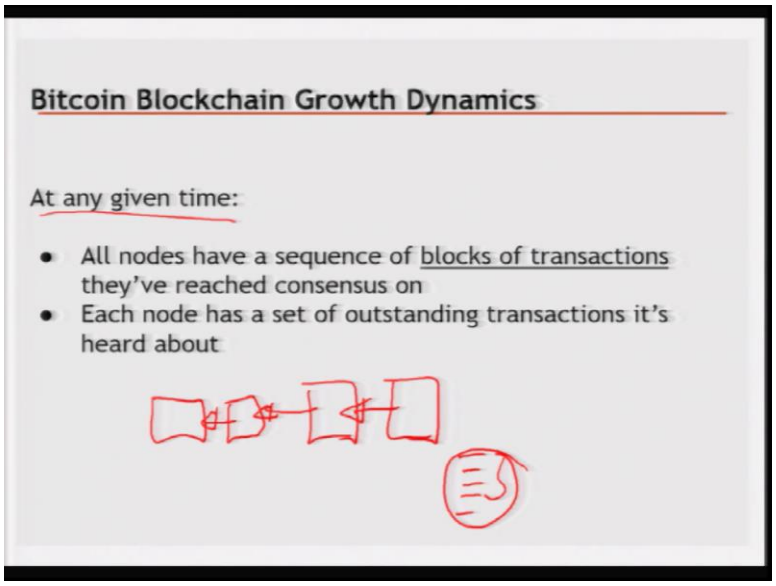
So how the peer to peer system works in blockchain in Bitcoin is that whenever you want to pay to your friend Ravi, some Bitcoin, you broadcast the transaction to every node of the network, saying that this is going to be Ravi's address, which is a hash of Ravi's public key, and you have to sign it. And you have to tell your public key also. And then you broadcast it to the entire network. Now, how the network broadcasted to reach all corners, is a matter of the protocol that is underneath the Bitcoin protocol.

It could be a gossip protocol. It could be other ways of broadcasting it when it can be a multicast and things like that. So now, one thing I want to say is that while you are paying Ravi, it is not a synchronized throw and catch. Ravi may not be connected at that time later on. So all the other nodes will check if your signature is valid. And also it will check whether you have in sufficient balance to pay Ravi.

Right the amount that you are telling you are going to pay, all the other nodes will check that and they will validate the transaction. And then eventually some nodes will create a block in which this transaction will be put in. And then eventually, this block will be, hopefully be part of the blockchain. Later on Ravi, when he connects, he will see that there is a transaction which gave him money.

And then after a while, we will talk about that, like how long after Ravi can assume that he can even spend that money. So, that is the way this peer to peer system is working.

(Refer Slide Time: 14:45)



Bitcoin Blockchain Growth Dynamics

At any given time:

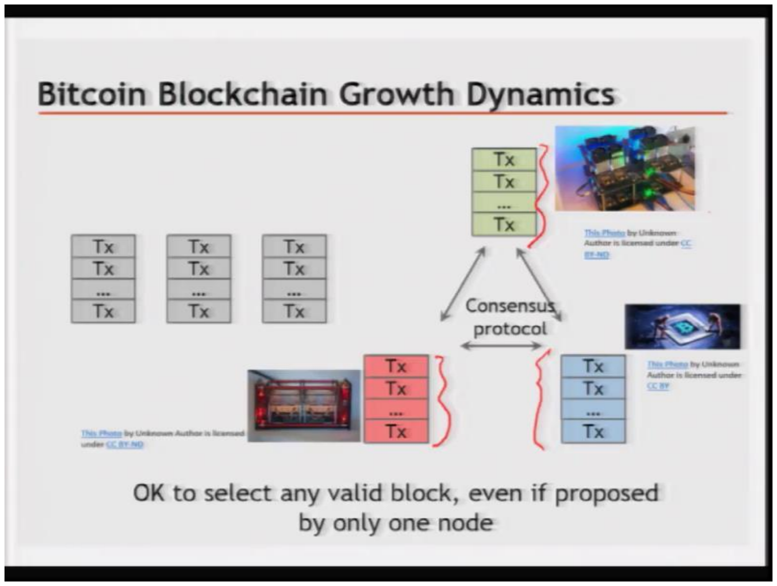
- All nodes have a sequence of blocks of transactions they've reached consensus on
- Each node has a set of outstanding transactions it's heard about

Now, let us look at the growth dynamics. So I have been talking about this for a while, but let us look at it more formally. At any given time, all the nodes that are maintaining a copy or replica of the blockchain will have a sequence of blocks on which the consensus has taken place. And there has been an agreement that these are the blocks that have been made permanent into the blockchain. So, every node will have some set of nodes that are already have gone through the consensus.

And then each node will have a set of transactions that have not been part of any of these blocks. And they have to decide which of these transactions they will want to put in a block. So, they can decide that I will put only this because the blocks have also a fixed size. So, they may not be able to put all the transactions that are still outstanding. So, these are the outstanding transactions, you have to select from them.

And then put in a new block and every node that is part of the process is doing the same thing. They have a sequence of blocks and that are already consented on and then they have a whole set of outstanding transactions.

(Refer Slide Time: 15:58)



Bitcoin Blockchain Growth Dynamics

OK to select any valid block, even if proposed by only one node

When you have this outstanding transition let us say in this example, there are 3 different miners. And they are using, of course, the mining requests a hash puzzle solving, so they have very high computing power, and so on. So this guy decides this set of transactions is going to be in the next block. This guy decides that these are the set of transactions in the next block. And this guy decides that these are the set of transactions in the next block.

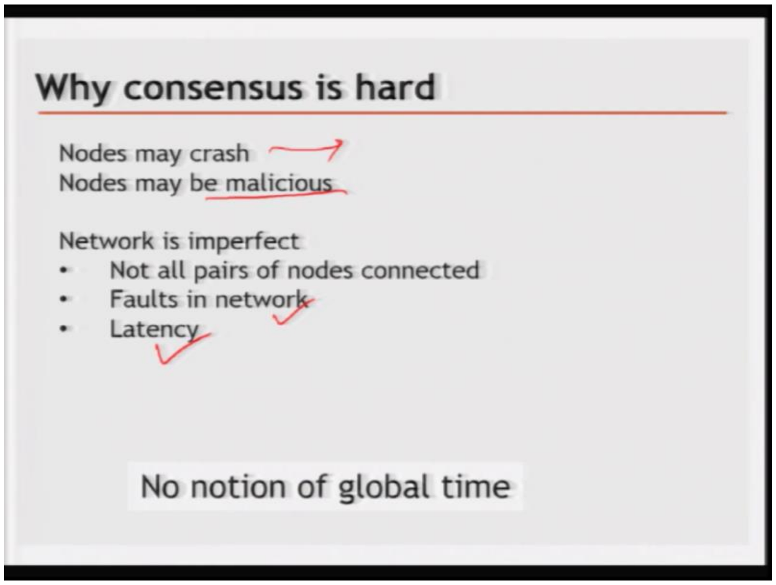
And this there may be many reasons why these 3 are different. One possibility is that the transaction as their broadcast, they take some time to reach all parts of the network. So by the time this guy collected some transactions, some of the transactions have not even reached that node. So therefore, this node, he does not even know the existence of a transaction. This node can also decide that I have this much space, so many kilobytes in a block.

I have too many transactions. I will randomly select some of them. And this guy also has the same issue so he might have heard of this transactions or gotten a copy of the transaction that this guy did not get. But at least he got that so he puts that in but on the other hand for the other transactions, he has randomly chosen from the outstanding transactions that will fit into his block. There may be other reasons, we will talk about that there might be an incentive reason why they choose certain transactions, which may not be fully random.

But in any case, all these blocks will look different. Maybe not maybe some of them will be exactly the same by virtue of probability, but there is no they are not talking to each other deciding which transactions go to the next block. So everybody will propose a block which may be slightly different. Now, when you propose a block, they and if you broadcast that this is my next proposed block, this block should be part of the blockchain.

If everybody does that, there will be a huge amount of data going in and then whoever every node that is keeping a copy of the of the blockchain will be confused because you will get thousands of different block proposals and he would not know which one to take so, there has to be some kind of consensus mechanism through which the next block has to be chosen. So let us see what happens.

(Refer Slide Time: 18:23)



Why consensus is hard

Nodes may crash

Nodes may be malicious

Network is imperfect

- Not all pairs of nodes connected
- Faults in network
- Latency

No notion of global time

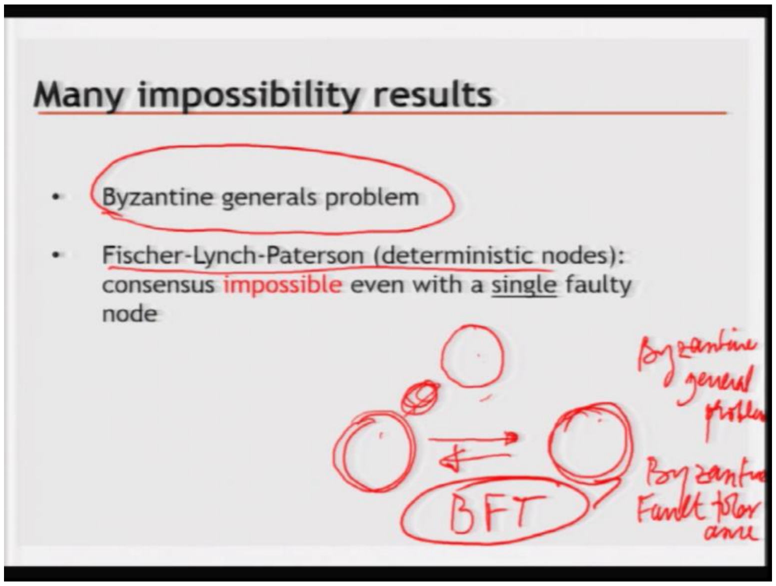
Now, as I said before, that we there may be a crash fault in some nodes while they are participating in the consensus process or some nodes may behave maliciously on top of that, the network has imperfections. So, not all pairs of nodes are directly connected of course, so they there are long paths between them or sometimes there is a segmentation in the network because of various reasons. So, not all transactions are reaching or all block broadcasted reaching all nodes very instantaneously, they might take time.

So there is latency and then the network may have faults because of that maybe routing is taking place and so, on top of that, one could say that well, if everybody brought custom block, I will look at the time stamp on them. And then I will decide which one has the lowest timestamp and I will take it but that that is also problematic because the notion of time at the various nodes which are geographically distributed is not unique.

So and then the clocks have to are not synchronized, there is no global clock maintained to do this, if there was a global clock, that is every computer has the same clock, then they could have actually done it much more easily by saying that the lowest timestamp one will be taken. But now the timestamps will be not trustworthy because none of the nodes are at different time zones first of all, but even if you adjust for the time zone.

Everybody’s let us say, expresses their time as a GMT or UNIX time even then there is a problem because clocks some clocks are slower because of various drifts and skews and other clocks may be faster. So, unless you maintain a global time using GPS clock or some other protocol for time synchronization, this will not work. So, time stamp is not a solution to this problem.

(Refer Slide Time: 20:26)



Many impossibility results

- Byzantine generals problem
- Fischer-Lynch-Paterson (deterministic nodes): consensus impossible even with a single faulty node

Byzantine general problem
BFT
Byzantine Fault tolerance

But while we go and look at the literature on consensus, we found that this problem of consensus is also called the Byzantine generals problem. Later in the course we will actually discuss this in much more detail. But the idea here is that if you have target enemy site and there are 2 Garrison's which are disconnected from each other, except they can send messages to each other and they know they cannot win this, the attacking this.

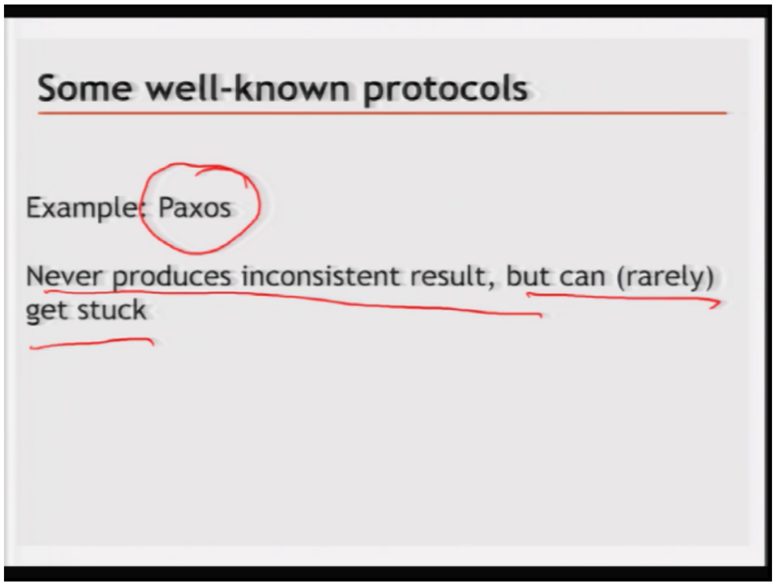
And when this unless both of them actually attack simultaneously from these 2 sides. If one of them attacks and then the other does not, then whoever attacks will be killed completely. So then we have a consensus problem that we have 2 Garrison's with 2 generals. And they have to decide whether they are going to attack or whether they are not going to attack. And either both of them have to not attack or retreat or both of them has to decide to attack.

So this is what is called Byzantine generals problem. Now, the problem here is that if all the generals are honest, and the messengers who were sent and back and forth are honest, then this problem is not a big deal. So, one general decides and then sends to this guy acknowledges and then both are going to attack or both are going to retreat. The problem is that if you assume that one of the general could be malicious or the messengers could be actually changing the messages, then this may actually fail.

So, in that case, they might, this guy might think that the consensus is to attack and this guy is actually think that consensus is to do retreat and therefore, attack alone and then get decimated right. So, therefore, you have to have a consensus that such that the honest generals, opinion should eventually prevail, when the entire thing settles. So therefore, this problem has been a model for this consensus problem as Byzantine fault tolerance problem or called BFT.

So we will see a lot of this discussion and BFT later. So, but early result from 1979, I believe by this is also called FLP result. The FLP basically says that if all nodes run a deterministic algorithm, then consensus is impossible even when a single node can crash. So, this is a very bad news. So, what we are saying is that solving Byzantine generals setting consensus problem, or BFT, Byzantine fault tolerant consensus is impossible even when a single node can be faulty. So, that is an in our case, we cannot even guaranteed single node it could be many nodes that are going to be malicious or faulty so then then what do you do.

(Refer Slide Time: 23:57)



Some well-known protocols

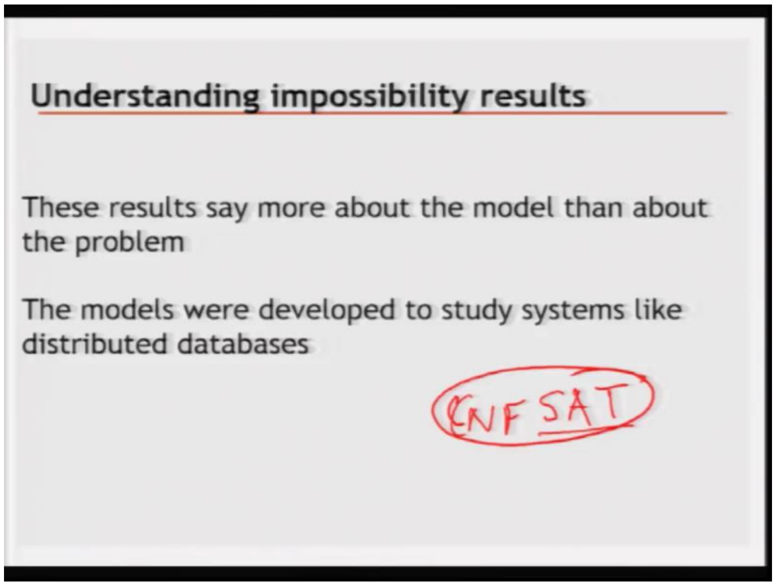
Example: Paxos

Never produces inconsistent result, but can (rarely) get stuck

But there are practical algorithms such as Paxos. So we will also later talk about this. But Paxos is a Byzantine fault tolerant consensus protocol. And we the way Paxos works is that well it never produces an inconsistent result that is the some nodes are saying yes and some nodes are saying no, but sometimes it can get stuck and never terminate. So, remember we said that a consensus problem protocol should always terminate.

And they should always give a result and all the good nodes are non-malicious nodes should arrive at the same result. And that result should be proposed by one of the good nodes. So that was the requirement of a consensus protocol. Now, Paxos does everything else except that it may not terminate sometimes. So you may get stuck but there are other ways to solve this problem.

(Refer Slide Time: 25:02)



Understanding impossibility results

These results say more about the model than about the problem

The models were developed to study systems like distributed databases

So, one thing that is interesting about impossibility results or NP hardness results or any kind of complexity lower bound result is that it is always the worst case scenario for example, those of you know NP hardness so satisfiability sat is CNF SAT is known as NP hard NP complete problem. And it turns out many sat instances are not difficult to solve, but there are very specific types of sad problems which require you know, exponential time.

So, therefore, what, what we can actually say is that impossibility result depends on the very corner case or worst case also, the result of impossibility here in case of this FLP result is more about your assumptions in how the model is supposed to work and what their behavior is, for example, they assume that the model requires that every node does a deterministic activity. So there has been Byzantine fault tolerant algorithms, which work with almost probability one, which are probabilistic or randomized algorithms.

These models were developed to study systems like distributed databases. And now, what we will see as we come back is that this FLP result does not, you know, stop us from actually getting consensus probabilistically in the blockchain So, we will see that the probabilistically the consensus works in blockchain.